Speeding Up Heuristic Function Synthesis via Extending the Formula Grammar

Sergio Poo Hernandez, Vadim Bulitko
Department of Computing Science, University of Alberta, Edmonton, AB, T6G 2E8, Canada
{pooherna, bulitko}@ualberta.ca

Abstract

Heuristic search algorithms have long been used in video-game AI for unit navigation and planning. The quality of the solution they produce depends substantially on the quality of the heuristic function they use. Recent work automatically synthesized human-readable heuristic functions for a given pathfinding map. This enables tailoring a heuristic to the map but is expensive since each map requires an independent synthesis run. In this paper we propose and evaluate re-using elements of heuristics synthesized for one map in synthesizing heuristics for another map. We do so by adding parts of a synthesized heuristic back to the grammar that defines the space of heuristic functions for the synthesis.

Introduction & Problem Formulation

Performance of heuristic search algorithms depends substantially on accuracy of the heuristic functions they are used with. In particular, real-time heuristic search agents tend to repeatedly revisit states when their heuristic function is misleading. This behaviour, known as scrubbing (Sturtevant and Bulitko 2016), hinders applicability of real-time heuristic search (RTHS) in video games — a domain which otherwise would be a great fit to real-time search. Indeed, RTHS algorithms such as LRTA* (Korf 1990) with standard heuristics tend to produce solution paths much longer than necessary.

In this paper we extend a recent line of research which represented heuristics as algebraic formulae and then generated (or synthesized) them automatically for a given video-game map (Bulitko 2020). We show how the synthesis process can be sped up by automatically forming building blocks in the formula space. We do so by extending the grammar defining the space of the heuristic formulae with elements derived from synthesis runs. Empirical evaluation shows that the extended grammar is beneficial on a novel set of video-game maps, not seen during the extension process.

Related Work

Memory-based heuristics have been developed for video game pathfinding (Björnsson and Halldórsson 2006; Sturtevant et al. 2009). Such approaches pre-compute a high-performance heuristic for a specific map. The resulting data structure has a substantial memory footprint and is generally not portable (i.e., cannot be used on another map).

More recent work by Bulitko (2020) automatically synthesized heuristics in the form of human-readable arithmetic formulae. While the memory footprint is negligible, the resulting formulae were found to be map specific thus lacking portability to other maps. Running formula synthesis for each map is expensive, in part because each synthesis starts from scratch, potentially re-discovering the same useful building blocks for the formulae.

Our Approach

In this paper we extend the automatic formulae synthesis by Bulitko (2020) as follows. We retain their representation of heuristic functions for RTHS as algebraic formulae. Each formula is compact, has a negligible memory footprint and human-readable. It defines the initial heuristic function for an LRTA*-like algorithm in pathfinding on grid maps. For each map the synthesis process searches the space of formulae induced by a context-free grammar and returns a formula which captures information about the map and guides real-time heuristic search effectively. Thus the performance measure being optimized is the average suboptimality of solutions produced by a fixed RTHS algorithm guided by a synthesized heuristic.

The space of formulae defined by the grammar is large thereby making such *per-map* synthesis process computationally expensive. In this paper we attempt to re-use results of synthesis across different maps. We do so by decomposing formulae synthesized on one map into subformulae and adding them to the grammar. The extended grammar then defines a new space of formulae which now incorporates elements of previously synthesized solutions without having to re-discover them from scratch.

The idea is similar in spirit to classical work on solution abstraction and re-use (Laird, Rosenbloom, and Newell 1986; Sutton, Precup, and Singh 1999; Botea et al. 2005) but is used for formula-based heuristic synthesis in real-time heuristic search for the first time.

We synthesize the best formula for an initial set of maps

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

using the following *baseline grammar* :

$$\begin{array}{lcl}
F & \to & T \mid U \mid B \\
T & \to & x_1 \mid x_2 \mid y_1 \mid y_2 \mid \Delta x \mid \Delta y \mid C \\
C & \to & 1 \mid 2 \mid \ldots \mid 6 \\
U & \to & \sqrt{F} \mid |F| \mid -F \mid F^2 \\
B & \to & F + F \mid F - F \mid F \times F \mid \frac{F}{F} \mid \max\{F, F\} \mid \min\{F, F\}
\end{array}$$

Here x_2, y_2 are goal coordinates on the grid map, x_1, y_1 are coordinates of the state whose heuristic value is being computed and $\Delta x = |x_1 - x_2|$ and $\Delta y = |y_1 - y_2|$.

After synthesizing a formula for each map using a problem set, we consider all possible subformulae that comprise the formula tree of each map. Each of the subformulae is then evaluated on another set of pathfinding problems. If the resulting path suboptimality is within a user-specified range of the initial formula's suboptimality the subformula is added to the grammar as a new terminal symbol under T. We do not consider terminal nodes (T in the grammar) as subformulae.

As an illustration, suppose the following heuristic function for map is synthesized using the baseline grammar: $-\left(-\sqrt{\min\left\{\Delta y + \Delta x, |\sqrt{y_2}|\right\}}\right)$. This formula is then added to the grammar along with its subformulae: $-\sqrt{\min\left\{\Delta y + \Delta x, |\sqrt{y_2}|\right\}}$, $\sqrt{\min\left\{\Delta y + \Delta x, |\sqrt{y_2}|\right\}}$, $\min\left\{\Delta y + \Delta x, |\sqrt{y_2}|\right\}$, $\Delta y + \Delta x$, $|\sqrt{y_2}|$, $\sqrt{y_2}$.

Synthesizing a heuristic function with the extended grammar on a different map can then yield a new formula $\sqrt{3 \cdot (\Delta y + \Delta x)}$ which includes the new element $\Delta y + \Delta x$ in the grammar.

Empirical Evaluation

The common video-game map repository (Sturtevant 2012) contains maps of different size and vastly different search complexity. To control search complexity we procedurally generated 20, 50×50-grid-cell maps using genetic algorithms. Each map was evolved to be complex enough that LRTA* with Manhattan distance had an average solution suboptimality of at least 25 (i.e., produced solutions 25 times longer than optimal on average).

We used ten of such maps for grammar extension. To synthesize the formulae we used a set of all possible problems generated by combining three starts and three goals on the maps for a total of nine problems. To evaluate the synthesized formulae we used the problem set generated from selecting 10 sets of starts and goals from the same maps. For the grammar extension, once the formula for each map was synthesized we broke it down into all possible subformulae and evaluated each on the map used to generate them. If a subformula suboptimality did not exceed the suboptimality of the host formula by more than 10% we added the subformula to the grammar.

We then synthesized the formulae for the remaing ten maps. We ran synthesis with the baseline grammar and with the extended gramar. We ran three independent trials of the synthesis process and plotted average solution suboptimality over all trials and all maps in Figure 1, top. Better heuristics are synthesized over time, resulting in lower suboptimality. However, the improvement is more substantial with the extended grammar than with the baseline grammar.

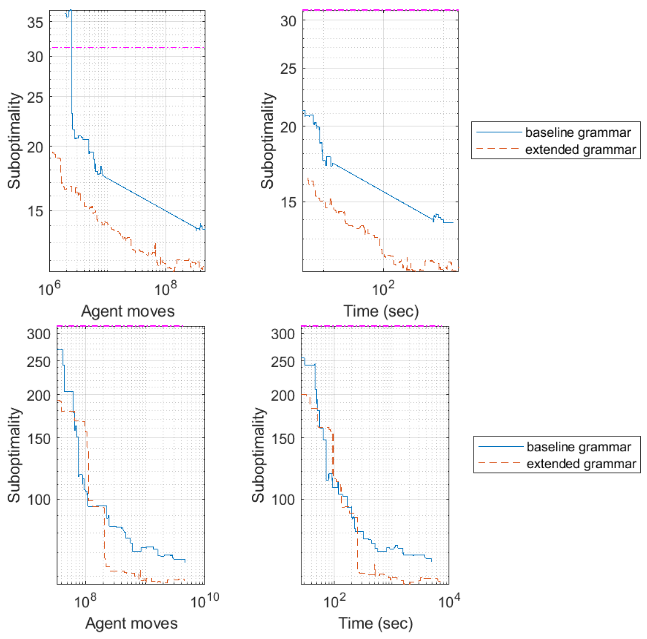

Figure 1: Top: experiments on 50×50 evolved maps. Bottom: experiments on video-game maps. Both sets plot solution suboptimality as a function of synthesis progression expressed in agent moves (left column) and wall-clock time (right column). The pink line is the solution suboptimality with the standard human-designed heuristic: Manhattan distance.

It is not surprising that adding subformulae synthesized for a set of maps to the grammar accelerates the synthesis process. In our next experiment we investigated if the extended grammar can accelerate synthesis on a substantially *different* class of maps. To do so we used the grammar extended with formulae synthesized for 50×50 evolved maps on larger, human-designed video-game maps from *Dragon Age: Origins (DAO)* (Sturtevant 2012). The bottom graphs in Figure 1 suggests that the extended grammar is indeed portable and facilitates a faster synthesis on a novel set of maps.

Conclusions

This work extended the recent line of research on automatically synthesizing heuristic functions for real-time heuristic search as compact, human-readable algebraic formulae. The synthesis process conducts a search in the space of formulae defined by a context-free grammar. In this paper we demonstrated how adding high-performing synthesized formulae and their parts back to the grammar accelerated subsequent synthesis even when used on novel maps.

References

Björnsson, Y.; and Halldórsson, K. 2006. Improved Heuristics for Optimal Path-finding on Game Maps. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment Conference*, 9–14.

Botea, A.; Enzenberger, M.; Müller, M.; and Schaeffer, J. 2005. Macro-FF: Improving AI planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research* 24: 581–621.

Bulitko, V. 2020. Evolving Initial Heuristic Functions for Agent-Centered Heuristic Search. In *Proceedings of IEEE Conference on Games*, 534–541.

Korf, R. 1990. Real-time heuristic search. *Artificial Intelligence* 42(2–3): 189–211.

Laird, J. E.; Rosenbloom, P. S.; and Newell, A. 1986. Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning* 1(1): 11–46.

Sturtevant, N.; and Bulitko, V. 2016. Scrubbing During Learning In Real-time Heuristic Search. *Journal of Artificial Intelligence Research* 57: 307–343.

Sturtevant, N. R. 2012. Benchmarks for Grid-Based Pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2): 144 – 148.

Sturtevant, N. R.; Felner, A.; Barrer, M.; Schaeffer, J.; and Burch, N. 2009. Memory-Based Heuristics for Explicit State Spaces. In *Proceedings of International Joint Conference on Artificial Intelligence*, 609–614.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1-2): 181–211.